Lesson 8

1. Introduction

So far you have worked with feature classes, rasters, records, cursors and text files. In this lesson you will have the opportunity to work with map documents. Python's ability to work with map documents is somewhat limited. In order to build customized tools to be used within a document you would need to use ArcObjects, a completely separate library and a different coding environment (Microsoft Visual Studio). Still, you can perform some useful tasks on map documents with Python.

2. Check list

Before we go any further with the lesson, here are the tasks you need to complete for this lesson:

- ☐ Read chapter 10 from the Zandbergen textbook
- ☐ Read the lesson notes
- ☐ Complete the practice exercises
- ☐ Write the pseudo code for your project
- ☐ Complete and submit your homework exercises

3. Working with map documents

The first step in working with map documents is to open a map document. When you open a map document ArcMap will not launch, but your script will have access to the document you opened and the document will be locked from other users. Here is how you can open and close a map document:

```
import arcpy

mapDoc = arcpy.mapping.MapDocument("C:/temp/Lesson8.mxd")
print "Got mapdoc"
del mapDoc
print "Closed mapdoc"
```

Issuing the delete command does not delete the map document file in your file system, it only deletes the map document object that you created within Python.

If you will recall from a previous lesson you were able to import the spatial analyst library, which allowed you to not use the arcpy.sa prefix to tools and functions that belonged to it. You can do the same with the mapping library by adding:

```
from arcpy.mapping import *
```

That will allow you to create the map document object like this instead:

```
mapDoc = MapDocument("C:/temp/Lesson8.mxd")
```

If you make any changes to the document you will need to save the modified document, which can be accomplished by the save method:

```
mapDoc.save()
```

Alterinatively you can use the saveACopy method, which allows you to specify a different name for the map document, and will leave the original document intact. It is a great idea to use the saveACopy method especially during testing, as it allows you to keep the original and test your changes by creating a separate document. SaveACopy is used in a similar manner to the save method, other than it needs the new document path and name:

```
mapDoc.saveACopy("C:/temp/Lesson8a.mxd")
```

One thing you cannot do is create a map document from scratch. You can open an existing document, modify its elements and save it as a map document with a different name, but you cannot create a new map document.

3.1. Accessing Map Document Properties

Once you have created a map document object from a path there are a number of document properties you can obtain. For example:

```
print mapDoc.author
print mapDoc.description
```

You will notice that you did not get a print out for the description if you are using the class data. That is because no description was set. Both the author and description properties are read/write, so you can set those properties too:

mapDoc.description = "This is a lesson 8 mxd document."

You can either use Python to confirm the description was written, or open the map document and go to File – Map Document Properties to look at the description. If you are

using saveACopy method to save the document and giving it a new name, make sure you are accessing the document you saved.

3.2. Working with Data Frames and Layers

Map documents are composed of data frames, which in turn contain layers. To obtain the list of data frames in a map document you will need to use the ListDataFrames method. To list layers you will use the ListLayers method:

```
dataFrames = ListDataFrames(mapDoc)

for frame in dataFrames:
    print frame.name

layers = ListLayers(mapDoc)

for layer in layers:
    print layer.name
```

Both the data frames and the layers have their own properties you can check and set. You can obtain a comprehensive alphabetized list of classes, with properties and methods, from the mapping library, including those for data frames and layers: http://resources.arcgis.com/en/help/main/10.2/index.html#/Alphabetical_list_of_arcpy_mapping_classes/00s30000000t000000/.

As the help document shows, you can check units, extents, spatialReference, and scale (among others) for the data frames.

Once you obtain a list of data frames and/or layers you can use an index to refer to them. For example you would print the name of the first data frame like this:

```
print dataFrames[0].name
```

The ListLayers method will list all layers in the map document unless a data frame is listed as well. The format of the ListLayers method is:

ListLayers (map_document_or_layer, {wildcard}, {data_frame})

The wildcard can be used if you want to filter only layers with a certain name. If you want to return layers regardless of name, you can set an empty string in place of that parameter. So, you would obtain a list of layers for the first data frame like this:

listLayers = ListLayers(mapDoc, "", dataFrames[0])

There are a number of useful properties and methods you can use with layers. You can obtain a complete list and description at: http://resources.arcgis.com/en/help/main/10.2/index.html#//00s300000008000000. You can for example, find out if the layer is a raster layer, a feature layer or a group layer. You can find out if the data source is broken. You can also change the path to the data source, and/or set if the layer is visible or not.

3.3. Creating a PDF Document

Arcpy.mapping library can be very powerful and helpful when automating creation of maps and map books. We will cover a simple scenario in this lesson, but if you are needing to create a lot of elaborate map books then using data driven pages along with arcpy.mapping library will be a great asset to you. If you have not used data driven pages before, here are a few ESRI help page resources to help you get starterd: http://resources.arcgis.com/en/help/main/10.2/index.html#//00s90000003m000000, http://resources.arcgis.com/en/help/main/10.2/index.html#//00s90000003n000000, http://resources.arcgis.com/en/help/main/10.2/index.html#//00s900000038000000.

Doing an export to a PDF from a map document is easy, especially if you do not need to make any modifications to the map document.
Arcpy.mapping will allow you to set up the legend, frame around the map, title and scale bar. You can of course set all those elements up ahead of time, but you could also choose to alter them with your script if necessary.

You can export a map to a PDF using the ExportTOPDF method. Here is what your code would look like:

```
import arcpy
from arcpy.mapping import *

mapDoc = MapDocument("C:/temp/Lesson8.mxd")
dataFrames = ListDataFrames(mapDoc)
dataFrame = dataFrames[0]

pdfPathdf = "C:/temp/lesson8df.pdf"
pdfPathpl = "C:/temp/lesson8pl.pdf"


ExportToPDF (mapDoc, pdfPathdf, dataFrame, df_export_width =
1200, df_export_height = 800)
```

```
ExportToPDF (mapDoc, pdfPathpl, "PAGE_LAYOUT",
df_export_width = 1200, df_export_height = 800)

print "The pdf has been generated at " + pdfPath

del mapDoc
```

An important detail to make a note of here is the third parameter to the ExportToPDF tool. There are two different ExportToPDF lines listed in the script above. All the parameters are the same other than the export path and the third parameter. In the first case the third parameter is a data frame. In the second case it is the constant "PAGE_LAYOUT". When you open the resulting PDFs you will see that they are different. The PDF that was exported with the constant shows the layout view, whereas the PDF created using the data frame as the third parameter shows the data view of the current data frame.

Arcpy also provides tools to merge PDF documents. This is expecially useful if you are needing to create map books. You could use a set of MXD documents to provide the different components of the map books, such as a title page, an index page and individual data pages. After you use the ExportToPDF tool, you can then merge the different components. For example you can add the following lines to the code from the previous script:

```
# Start edding components to the newly created PDF
finalPDFDoc.appendPages(pdfPathdf)
finalPDFDoc.appendPages(pdfPathpl)

#Save the PDF and clean up
finalPDFDoc.saveAndClose()
del finalPDFDoc
```

The previous script uses the map layout view elements (title, legend) that were already preset in the map document. You can modify these elements if you need to through Python. For example, if you wanted to change the title element, you would have to obtain the object first, and then change it. Before you could do any of it however, you would need to name the element in ArcMap. The title has to exist on a map, so if it is not there, create it. Right click on it and select properties. Once the property window is up, go to the Size and Position tab and type in a name in the Element Name text box. In Figure 1 the element has been named title. You need to apply the properties and save the map before Python will be able to take advantage of the field name.

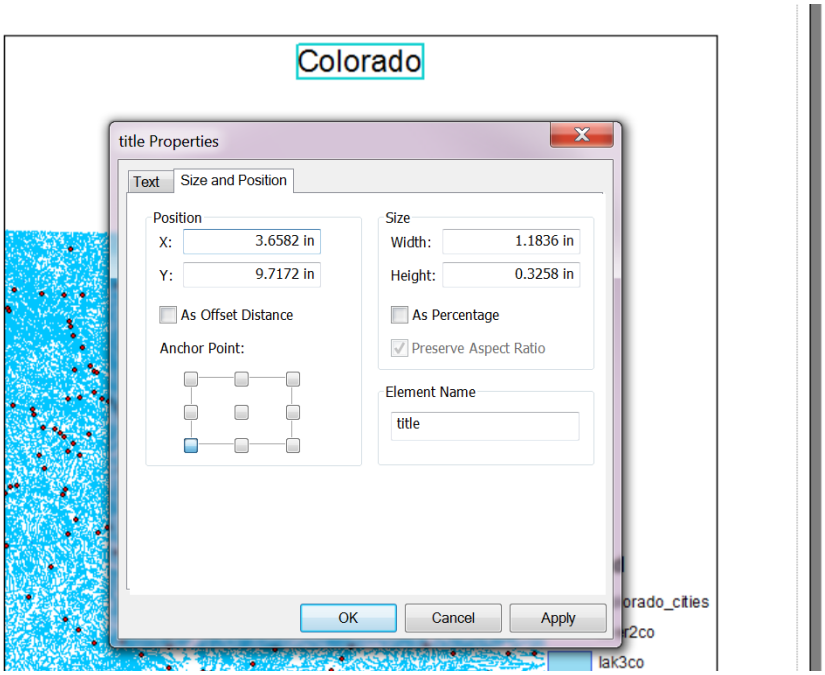

Figure 1

Once the field name is set up you can access the text element, and change its text property:

```
titleItem = ListLayoutElements(mapDoc, "TEXT_ELEMENT",
"title")[0]
titleItem.text = "This is an awsome map!!!"
```

The legend elements can be changed as well:
http://resources.arcgis.com/en/help/main/10.1/index.html#//00s300000041000000

3.4. Zooming

In a corporate environment people could modify map documents. Even if the elements stay the same there is always a possibility the map was zoomed and panned and does not look like you might want it to. The arcpy.mapping library provides tools to zoom to the full extent or selected features, or a custom extent.

Setting up the zoom for a full extent is simple. First of all you need to obtain the data frame as described above. Once you have the data frame object, and no features are selected you can issue this command:

```
dataFrame.zoomToSelectedFeatures()
```

Since nothing is selected, the zoomToSelectedFeatures method will zoom out to the full extent. Unfortunately, I found that even if you select a layer this tool tends to zoom to the full extent, rather than the selected feature. The section below describes how to zoom to a specific section.

Suppose you want to use the Jefferson county shapefile, and zoom just to those extents. First you would need to set the path, and then use the path to create a layer object. You can create a layer object by using the arcpy.mapping Layer method:

```
JeffersonCoPath = "C:/temp/Jefferson.shp"
JeffCo = Layer(JeffersonCoPath)
```

Once you have the layer, you can obtain the extent of the layer by using the getSelectedExtent method, and assign the results to the dataFrame.extent property:

```
dataFrame.extent = JeffCo.getSelectedExtent()
```

You could also change the scale a bit, to zoom out from the extent slightly. For example:

```
dataFrame.scale *= 2.5
```

Suppose you want to create a map for Douglas county, but you do not have a separate shape file for that county. Douglas county is one of the features in the Counties shape file, and you can select it from the file, and then zoom to the extent:

```
CountiesPath = "C:/temp/Counties.shp"


Counties = Layer(CountiesPath)
arcpy.SelectLayerByAttribute_management(Counties,
"NEW_SELECTION", "COUNTY = 'DOUGLAS'")
dataFrame.extent = Counties.getSelectedExtent()
```

4. Psuedo Code for the Final Project

In preparation for your final project, start thinking about the steps that your code needs to take to achieve the given task. Create a list of steps written in plain English, that you can later follow to create your script. Assume your project was to take the DEM for the state of Colorado, and create a hillshade and slope raster for Jefferson county. Your pseudo code should look something like this:

Create path variable for the DEM layer

Create path for Jefferson county shapefile
Clip the DEM layer with the Jefferson county shapefile
Create hillshade from the clipped DEM
Create slope from the clipped DEM

Creating pseudo code allows you to focus on the logic your script is supposed to follow, rather than focusing on the ins and outs of python syntax. Once you have your pseudo code you can use it in comment lines in your code to help you keep on track with your code.

You do not need to submit your pseudo code at this time, it will be a part of your final submission.

5. Practice problems

1. You just got hired and it is your first day at work. Your manager is leaving for a conference for a week, and not having enough time to bring you up to speed and give you meaningful assignments he asks you to go through a whole set of mxd files, which are separated into folders by continent, country and state, and change the author of all those documents to your name. He will be leaving in the afternoon and told you could take the rest of the week off as soon as you are done, thinking this will take a while as there are hundreds of documents (your dataset contains only a small subsection).

   Write a script that will accomplish this task, instead of you having to manually open each of these documents, which will also allow you to leave for the week before he leaves for the conference.

   To obtain the list of folders you will need to use the os library. Here is how you would obtain a list of folders located on the root of the C: drive

   ```
   import os

   dirList = os.listdir("C:/")
   print dirList
   ```

   You can use the arcpy ListFiles method to get a list of files once you are in a country directory.

   Use Lesson_8Data/Practice1 folder for this exercise. You will see that the folder has a continent folder. Within each continent folder there is a country folder, and within each country there are a list of state map documents. All the map documents are empty, but they will work fine for the purpose of this exercise.

   You will need to use nested loops – i.e. loops that run within loops.

2. There are some projects in your company with erroneous extents. Some data you have been using comes from external sources, and at times there are data entries with incorrect values causing your maps to have "blown" extents. You have been tasked to check all layers in the Practice2/Colorado.mxd document and list their extents, so problem layers can be identified and corrected.

   A couple of data sources will have broken links so no extents be available.

3. List all broken data sources in the Practice2/Colorado.mxd map document.

4. Set all raster layers in the first data frame to be invisible.

5. Due to some systems upgrades all shapefile data has been moved to a geodatabase. You task is to change all the layer paths from /Lesson8_Data to /Lesson8_Data/Practice5/Practice5.gdb. Use the /Lesson8_Data/Practice5/Colorado.mxd for this problem.

6. Homework Assignment

1. Use the Lesson8.mxd to create a PDF document that will contain a map for each county in the counties feature class. Change the title of each map to the name of the county being shown. Scale the map a bit so it shows an area just a bit larger than the actual extent.

2. Change the spatial reference of the data frame in Lesson8.mxd from UTM 13, NAD83 to un-projected NAD83. You can obtain the projection information from the NAD83.shp shape file.